Claims:

1. (Currently amended) A computer-implemented method of generating common intermediate language code for use in a framework, the method comprising:

Receiving at a computer a portion of JAVA™ language source code referencing, through a generic class syntax, one or more generic classes unspecified in a formal JAVA™ language specification, wherein:

each of the one or more generic classes refers to a first class configured to operate uniformly on ~~values~~ instances of a plurality of ~~different~~ types associated with the first class ~~and defined by a plurality of second classes~~;

the plurality of types are defined in the first class as an unconstrained type supporting a generic class type;

at least one of the one or more generic classes nests a second generic class as one of the plurality of types within the first class by associating declaration of instance of the second generic class with a defined first generic class; and

the generic class syntax is not specified in the formal JAVA™ language specification and identifies one instance of the plurality of types ~~second classes~~ by surrounding the one instance ~~of the plurality of second classes~~ with angular brackets following the first class; and

generating, through a first compiler different from a formal compiler complying with the formal JAVA™ language specification, language-neutral intermediate language

code representing the portion of JAVA™ language source code for execution at the computer and referencing the one or more generic classes.

2. (Previously Presented) A method as recited in claim 1 further comprising parsing the portion of the JAVA™ language source code into a parse tree representing the portion of the JAVA™ language source code before compiling the portion with the first class.

3. (Original) A method as recited in claim 2 further comprising nesting a constructed class of the first class in the parse tree.

4. (Currently amended) A method as recited in claim 1 further comprising:

generating a parse tree having a token referencing the first class and a token referencing the one instance ~~one of the plurality of second classes~~; and

semantically analyzing the parse tree to determine validity of semantics of the first class.

5. (Original) A method as recited in claim 4 wherein the semantically analyzing comprises determining whether operations applied to the first class are valid.

6. (Original) A method as recited in claim 1 further comprising generating metadata descriptive of the first class.

7. (Previously Presented) A method as recited in claim 6 further comprising storing the metadata with the language-neutral intermediate language code, whereby the language-neutral intermediate language code is used by an application program.

8. (Currently amended) A method as recited in claim 1 further comprising creating a compiled project including the language-neutral intermediate language code and metadata descriptive of the first class and the one instance ~~of the plurality of second classes~~.

9. (Original) A method as recited in claim 1 further comprising executing the language-neutral intermediate language code with a runtime engine.

10. (Canceled).

11. (Previously Presented) A method as recited in claim 1 wherein the framework is a .NET™ Framework.

12. (Previously Presented) A method as recited in claim 11 wherein the developing comprises authoring the portion of JAVA™ language source code with a VISUAL J# .NET™ application of the .NET™ Framework.

13-16. (Canceled).

17. (Currently amended) A method as recited in claim [[14]] 1 further comprising validating an operation on the instance of the generic class based on the defined generic class.

18. (Currently amended) A computer-readable medium having stored thereon computer-executable instructions for performing a method of compiling in a framework, the method comprising:

receiving a portion of JAVA™ language software including an instruction that references a generic class of a specified type through use of a generic class syntax, wherein:

the generic class is unspecified in a formal JAVA™ language specification and refers to a first class configured to operate uniformly on instances of a plurality of ~~values of different~~ types associated with the first class ~~and defined by a plurality of second classes~~;

the plurality of types are defined in the first class as an unconstrained type supporting a generic class type;

the generic class nests a second generic class as one of the plurality of types within the generic class by associating declaration of instance of the second generic class with the first class; and

the generic class syntax is not specified in the formal JAVA™ language specification and identifies one of the instances of the plurality of types ~~second classes~~ by surrounding the one instance ~~of the plurality of second classes~~ with angular brackets following the first class; and

creating a parse tree having a generic class identifier associated with the generic class and type identifier associated with the specified type; and

generating, through a first compiler other than a traditional compiler complying with the formal JAVA™ language specification, one or more intermediate language instructions representing the JAVA™ language instruction based on the parse tree .

19. (Currently amended) A computer-readable medium as recited in claim 18, wherein the method further comprises ~~comprising~~ translating the one or more intermediate language instructions into microprocessor-specific binary for execution by a computer.

20. (Currently amended) A computer-readable medium as recited in claim 18, wherein the method further comprises ~~comprising~~ validating the parse tree according to a generic class definition associated with the generic class.

21. (Original) A computer-readable medium as recited in claim 20, wherein validating the parse tree comprises determining whether an assignment applied to the instance of the generic class assigns an allowable type to the instance.

22. (Currently amended) A computer-readable medium as recited in claim 18, wherein the method further comprises ~~comprising~~ generating metadata associated with the generic class.

23. (Canceled).

24. (Currently amended) A computer- readable medium as recited in claim [[23]] 18, wherein the method further comprises nesting the second generic class and the second specified type at different levels in a hierarchy in the parse tree.

25-29. (Canceled).

30. (Currently amended) A computer-readable medium as recited in claim [[25]] 18, wherein the plurality of types ~~constructed class~~ comprises one of:

an integer type;

a float type;

a Stack type;

a Queue type; and

a Dictionary type.

31. (Currently amended) The method as recited in claim 36 further ~~A method of compiling in a framework, the method~~ comprising:

~~receiving a portion of source code in a first programming language for which one or more generic types are not specified in a formal definition of the first programming language, wherein:~~

~~each of the one or more generic types refers to a first type configured to operate uniformly on values of different types associated with the first class and defined by a plurality of second types;~~

~~each of the one or more generic types uses a generic type syntax not specified in the formal definition of the first programming language;~~

parsing the portion of source code into a parse tree comprising each instance of the one or more generic types in the portion of source code, wherein each instance of the one or more generic types comprises:

the first type; and

at least one instance of one of the plurality of second types associated with the first type~~; and~~

~~generating an intermediate representation of the parse tree representing the parse tree~~.

32. (Canceled).

33. (Previously Presented) The method as recited in claim 31 further comprising tokenizing the parse tree with a token corresponding to the one or more generic types.

34. (Previously Presented) The method as recited in claim 33 further comprising tokenizing the parse tree with at least one token corresponding to the at least on instance of one of the plurality of second types associated with the first type .

35. (Currently amended) The method as recited in claim [[31]] 36 wherein each of the one or more generic types is a .NET™ generic class.

36. (Currently amended) A computer-implemented method of generating microprocessor-executable code in a framework, the method comprising:

receiving at a computer a portion of source code written in a first programming language for which generic classes are unspecified, the portion of source code including a generic class declaration declaring a generic class, wherein:

the generic class refers to a first class configured to operate uniformly on values of different types associated with the first class and defined by a plurality of second classes;

the plurality of second class are defined in the first class as an unconstrained type supporting a generic class type;

the generic class nests a second generic class as one of the plurality of second classes within the generic class by associating declaration of instance of the second generic class with the first class;

the generic class uses a generic class syntax not specified in a formal specification of the first programming language;

the generic class declaration creates a constructed class of the generic class by associating a reference of one of the plurality of second classes with the generic class; and

generating a module having microprocessor-executable instructions corresponding to the constructed class based on the portion of source code, the module further having metadata describing the constructed class.

37. (Original) A method as recited in claim 36 wherein the microprocessor-executable instructions comprise intermediate language instructions.

38. (Original) A method as recited in claim 36 wherein the microprocessor-executable instructions comprise Microsoft® Intermediate Language instructions.

39. (Original) A method as recited in claim 36 wherein the metadata comprises at least one of:

a name of the constructed class;

visibility information indicating the visibility of the constructed class;

inheritance information indicating a class from which the constructed class derives;

interface information indicating one or more interfaces implemented by the constructed class;

method information indicating one or more methods implemented by the constructed class;

properties information indicating identifying at least one property exposed by the constructed class; and

events information indicating at least one event the constructed class provides.

40-42. (Canceled).

43. (Currently amended) A system as recited in claim 51, further comprising a validator configured to validate ~~method as recited in claim 40 further comprising validating the~~ type of the first class based on a definition of the first class.

44. (Currently amended) A system as recited in claim 43, wherein the validator validates ~~method as recited in claim 43 further comprising validating~~ an operation on the first class based on a definition of the first class.

45-46. (Canceled).

47. (Currently amended) A system as recited in claim 51, ~~method as recited in claim 40~~ wherein the first class is a Queue class.

48-50. (Canceled).

51. (Currently amended) A system for compiling in a framework, the system comprising:

a parser receiving JAVA™ language source code having an instruction referencing a generic class in a generic class syntax and specifying a type of the generic class, the parser further creating a parse tree from the JAVA™ language source code, the parse tree including a first node representing the generic class and a second node representing the specified type of the generic class, wherein:

the generic class refers to a first class configured to operate uniformly on values of different types associated with the first class and defined by a plurality of second classes;

the plurality of second class are defined in the first class as an unconstrained type supporting a generic class type;

the generic class nests a second generic class as one of the plurality of second classes within the generic class by associating declaration of instance of the second generic class with the first class;

the generic class syntax is unspecified in the formal language specification of JAVA™ programming language and supported in the framework; and

a code generator generating intermediate language code representing the JAVA™ language source code referencing the generic classes .

52. (Original) A system as recited in claim 51 further comprising:

a common intermediate language importer providing tokens associated with the generic class and the specified type of the generic class.

53. (Original) A system as recited in claim 51 further comprising a runtime engine executing the intermediate language code.

54. (Original) A system as recited in claim 51 further comprising a semantic analyzer analyzing the specified type to determine whether the specified type is an allowable type of the generic class.